

Real-time testing with Fuego

24 Oct 2018

Hiroataka MOTAI

OUTLINE

- Who I am
- Overview
- Related Tools
 - Automated Test Framework / Fuego
 - Real-time latency tool / cyclicttest
 - Tracing kernel feature / ftrace
- Issue
- Approach
 - Our Use Cases
- Conclusion and Future work

WHO I AM

- Hirotaka MOTAI
 - Software Researcher for embedded systems of Mitsubishi Electric Corp.
- We have collaborated with LF projects.
 - LTSI: Long Term Support Initiative
 - AGL : Automotive Grade Linux
 - FUEGO: Test framework
 - Specifically designed for testing Embedded Linux



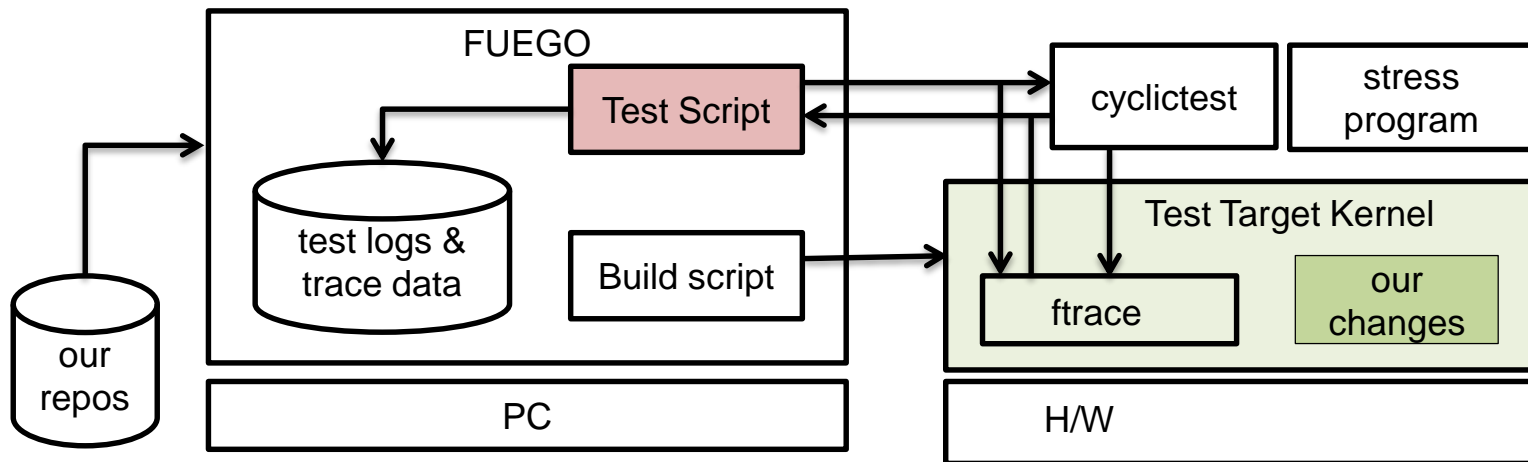
- Who I am
- **Overview**
- Related Tools
 - Automated Test Framework / Fuego
 - Real-time latency tool / cyclicttest
 - Tracing kernel feature / ftrace
- Issue
- Approach
 - Our Use Cases
- Conclusion and Future work

OVERVIEW

- Linux can be adapted to various embedded devices, even though they need a hard real-time response.
- We need tons of time to ensure adequate real-time performance.
 - Real-time applications need to satisfy timing constraints.
 - We have to avoid kernel changes which might cause long delays.

OVERVIEW

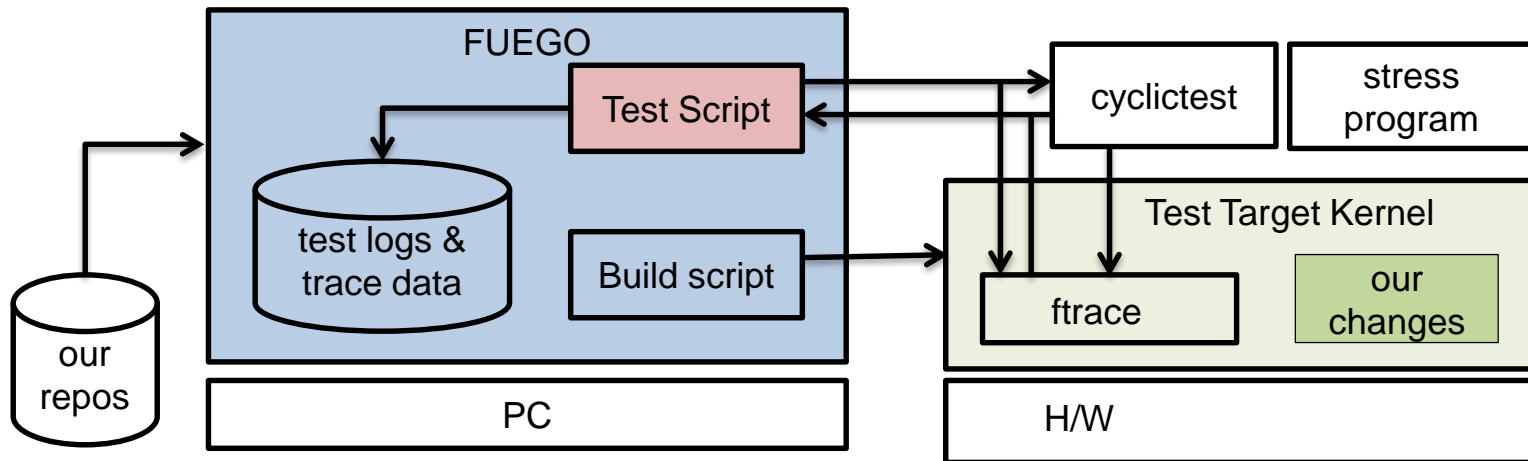
- I have developed a new test script on Fuego.
 - measure the real-time performance, plus get tracing.
 - get clues to isolate the problem whether it was caused by our changes or not.



- Who I am
- Overview
- Related Tools
 - Automated Test Framework / Fuego
 - Real-time latency tool / cyclicttest
 - Tracing kernel feature / ftrace
- Issue
- Approach
 - Our Use Cases
- Conclusion and Future work

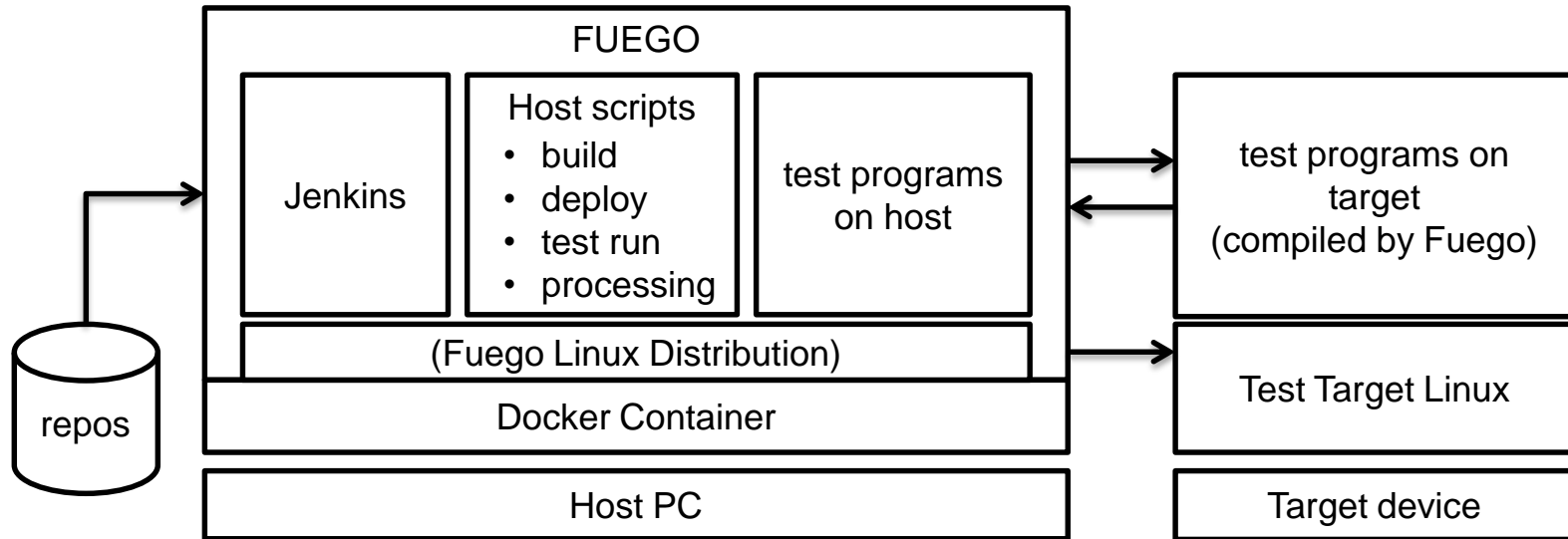
FUEGO

- Fuego is the automated test framework
 - created by LTSI project, based on Jenkins.
 - OSS: anyone can use and contribute!
 - AGL-JTA: AGL chose Fuego as standard test environment.



FUEGO

- Fuego = "test distribution + Jenkins + host scripts + pre-packaged tests" on container
- Fuego can do specific tests automatically that is triggered by software update.



- You can click to start manually and monitor tests on Jenkins.

The screenshot shows the Jenkins web interface for a project named 'minnow'. The browser address bar shows the URL '192.168.3.9:8080/fuego/view/minnow/'. The Jenkins logo and name are at the top left. A sidebar on the left contains links like 'New Item', 'People', 'Build History', 'Edit View', 'Delete View', and 'Manage Jenkins'. Below these are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status'.

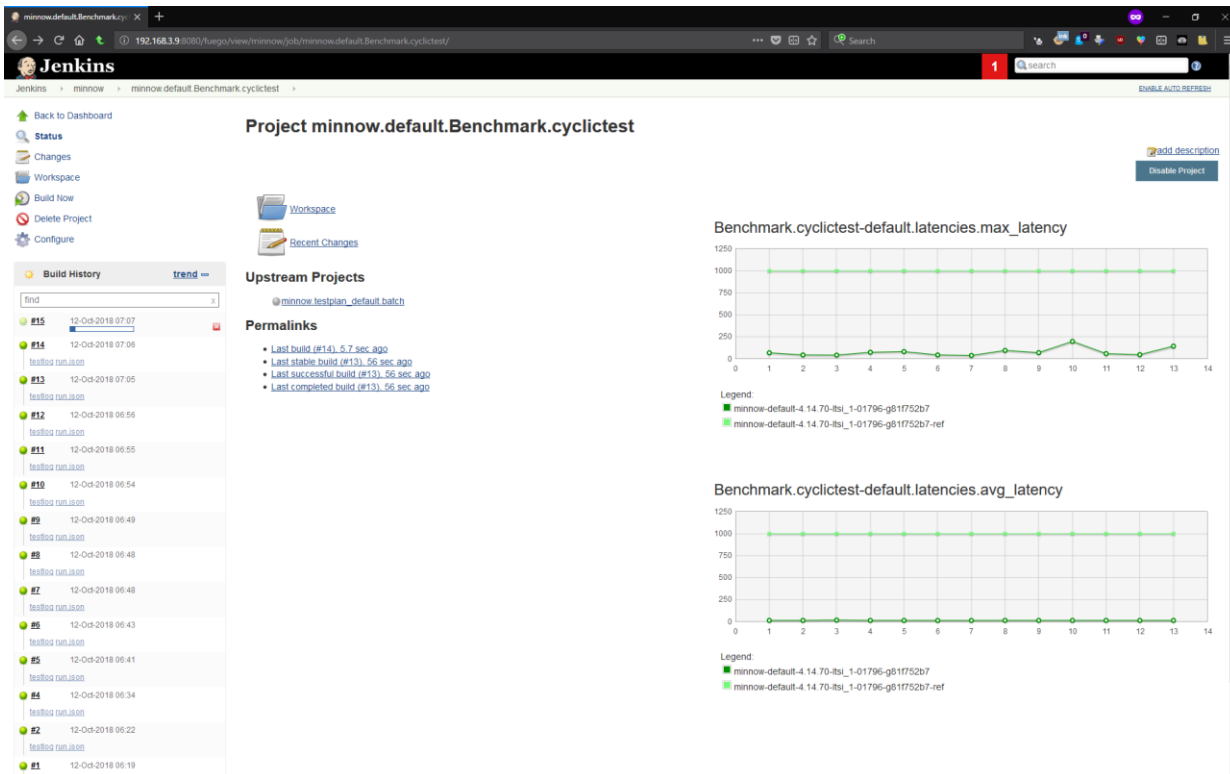
The main content area displays a table of benchmarks. The table has columns for 'S' (Status), 'W' (Weather/Icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. The 'S' column contains icons representing the status of each benchmark. The 'W' column contains sun icons. The 'Name' column lists various benchmarks like 'minnow.default.Benchmark.bonnie', 'minnow.default.Benchmark.cyclicttest', etc. The 'Last Success' and 'Last Failure' columns show the last time each benchmark was run successfully or failed. The 'Last Duration' column shows the time taken for each benchmark.

Annotations highlight specific features:

- click to start**: Points to a green circular button with a play icon in the rightmost column of the table.
- result**: Points to the 'S' column, which shows the status of each benchmark.
- monitor**: Points to a button in the 'Build Executor Status' section, which allows monitoring the progress of a build.

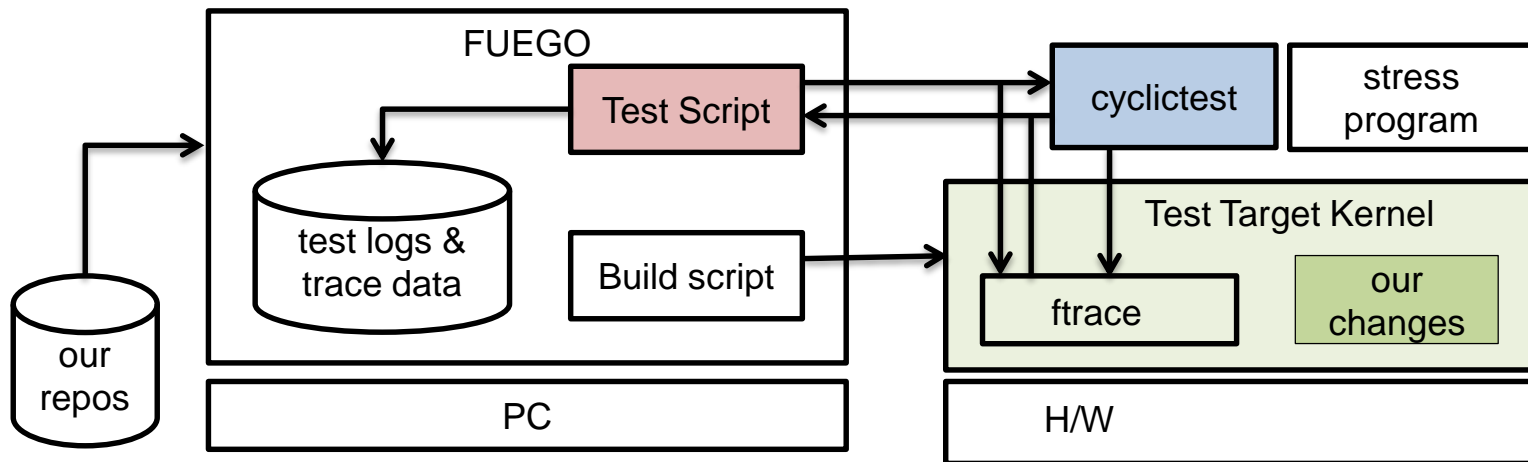
S	W	Name	Last Success	Last Failure	Last Duration
		minnow.default.Benchmark.bonnie	N/A	N/A	N/A
		minnow.default.Benchmark.cyclicttest	4 min 26 sec - #17	N/A	37 sec
		minnow.default.Benchmark.dbench4	N/A	N/A	N/A
		minnow.default.Benchmark.Dhrystone	N/A	N/A	N/A
		minnow.default.Benchmark.ebizzy	N/A	N/A	N/A
		minnow.default.Benchmark.ffsb	N/A	N/A	N/A
		minnow.default.Benchmark.fio	N/A	N/A	N/A
		minnow.default.Benchmark.GLMark	N/A	N/A	N/A
		minnow.default.Benchmark.gtkperf	N/A	N/A	N/A
		minnow.default.Benchmark.hackbench	N/A	N/A	N/A
		minnow.default.Benchmark.himeno	N/A	N/A	N/A
		minnow.default.Benchmark.interbench	N/A	N/A	N/A

- You can also check test results on Jenkins.



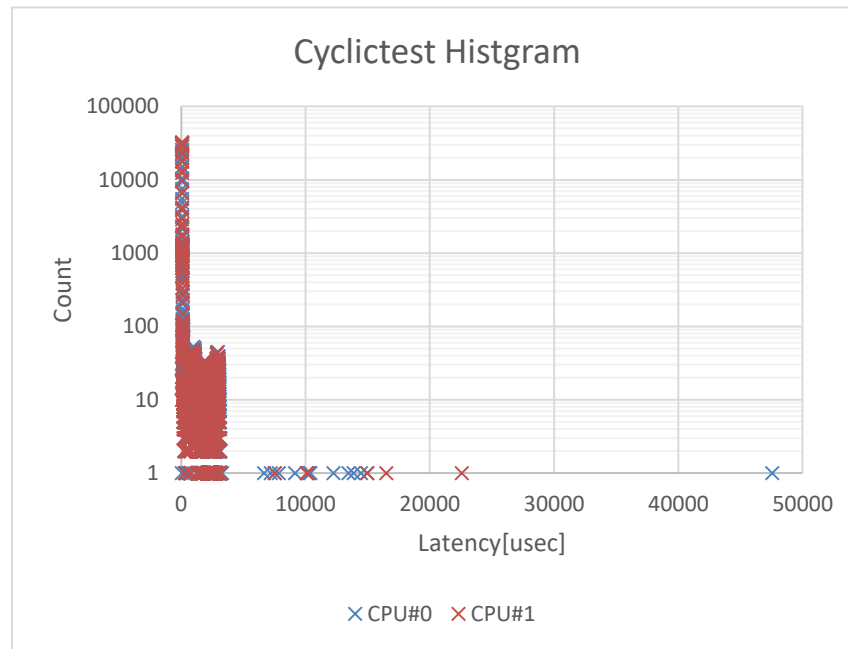
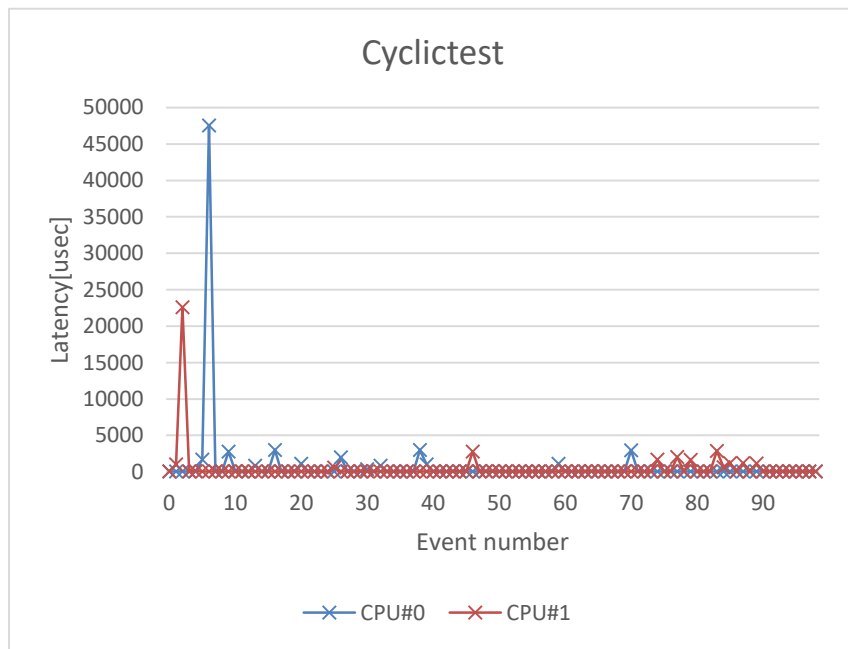
CYCLICTEST

- What is Cyclictest?
 - Benchmark tool for interval timer latency.
 - Refer to: <https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclictest>



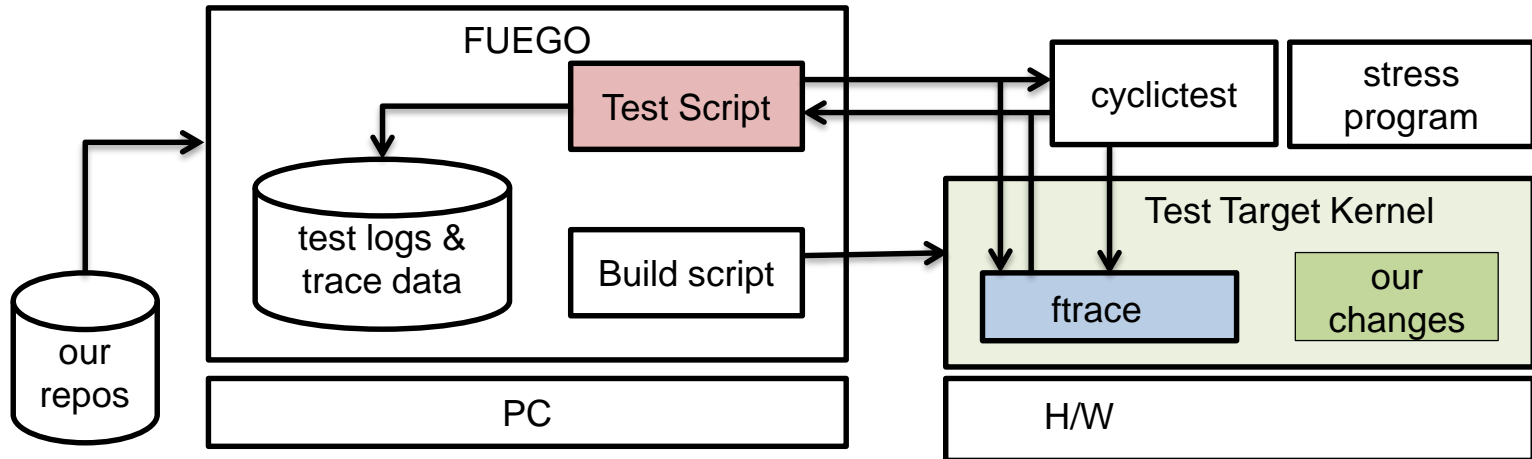
CYCLICTEST

- What is Cyclicttest?
 - Benchmark tool for interval timer latency.
 - Refer to: <https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclicttest>



FTRACE

- Ftrace have been in the kernel since Linux v2.6.27.
- Traces kernel without recompiling.
- Useful for data gathering, debugging, and performance tuning.
- Detailed in [Documentation/trace/index.rst](#)



- The Function Tracer

```
# tracer: function
#
#          _-----=> irqsoff
#          / _-----=> need_resched
#          | / _-----=> hardirq/softirq
#          || / _---=> preempt-depth
#          ||| /      delay
#          TASK-PID   CPU#   ||||   TIMESTAMP   FUNCTION
#          | |       |   |   ||||   |           |
stress-ng-shm-s-1257 [000] .... 7523.267555: vmacache_find <-find_vma
stress-ng_1h_pl-1194 [001] .... 7523.267556: mutex_unlock <-
                                   rb_simple_write
stress-ng-shm-s-1257 [000] .... 7523.267559: handle_mm_fault <-
                                   __do_page_fault

[snip]
```

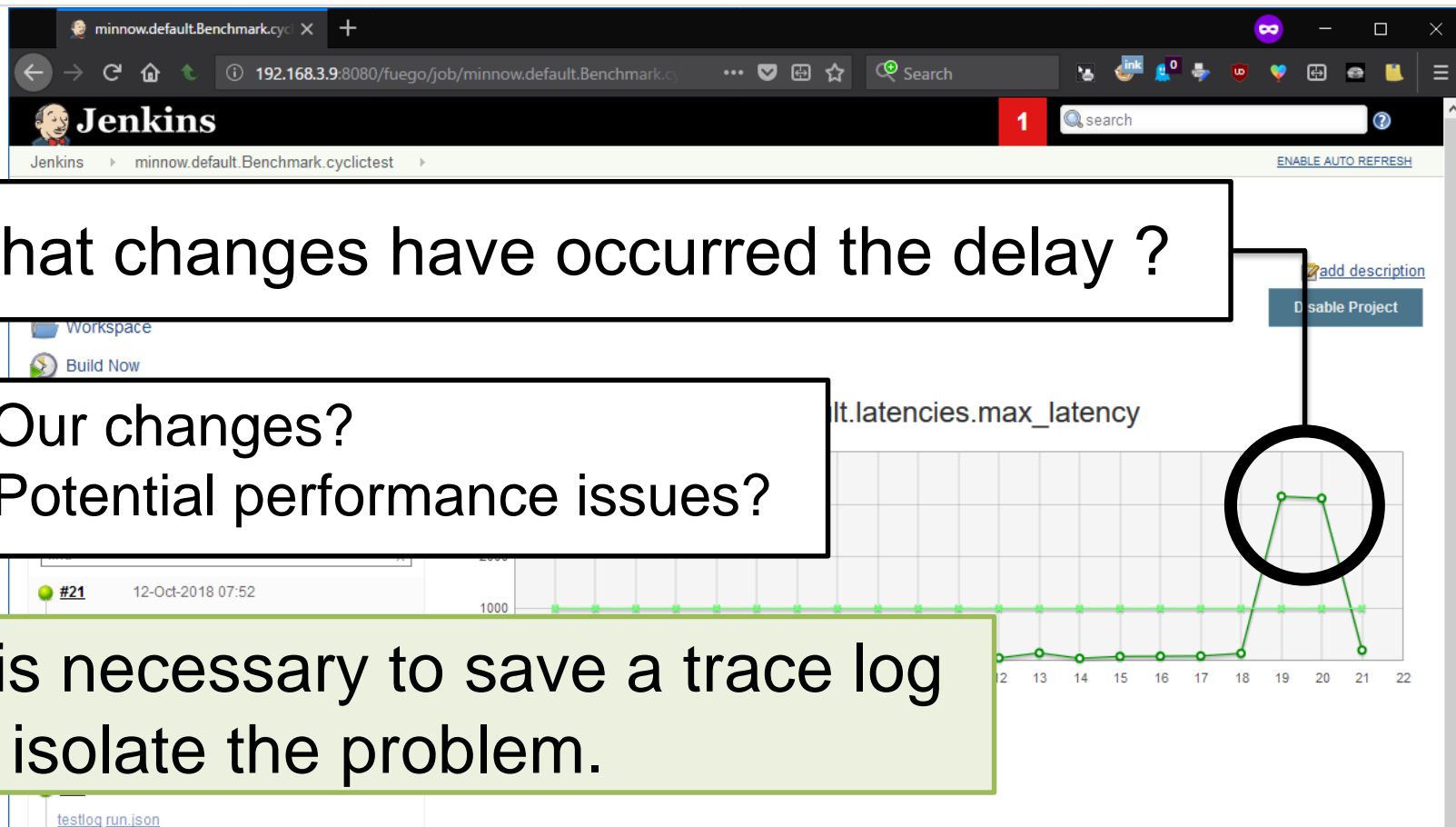
- Who I am
- Overview
- Related Tools
 - Automated Test Framework / Fuego
 - Real-time latency tool / cyclicttest
 - Tracing kernel feature / ftrace
- **Issue**
- Approach
 - Our Use Cases
- Conclusion and Future work

ISSUE

What changes have occurred the delay ?

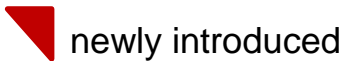
- Our changes?
- Potential performance issues?

It is necessary to save a trace log to isolate the problem.



- Who I am
- Overview
- Related Tools
 - Automated Test Framework / Fuego
 - Real-time latency tool / cyclicttest
 - Tracing kernel feature / ftrace
- Issue
- Approach
 - Our Use Cases
- Conclusion and Future work

- Do in a simple way.
 - Script-based Test driver program
 - 1: setup ftrace configurations
 - 2: run stress program
 - 3: run cyclicttest
 - 4: save a log with a ftrace data
- Cyclicttest option
 - "--breaktrace breaktime[us]"
can stop tracing
when latency > breaktime



STOP TRACING

- But, --breaktrace option will stop not only tracing but also testing, when $\text{diff} > \text{breaktime}$.

```

873         if (!stopped && t && (diff > tracelimit)) {
874             stopped++;
875             tracemark("hit threshold (%llu > %d)",
876                     (unsigned long) diff, tracelimit);
877             shutdown++;
878             pthread_mutex_lock(&break_thread_id_lock);
879             if (break_thread_id == 0)
880                 break_thread_id = stat->tid;
881             break_thread_value = diff;

```

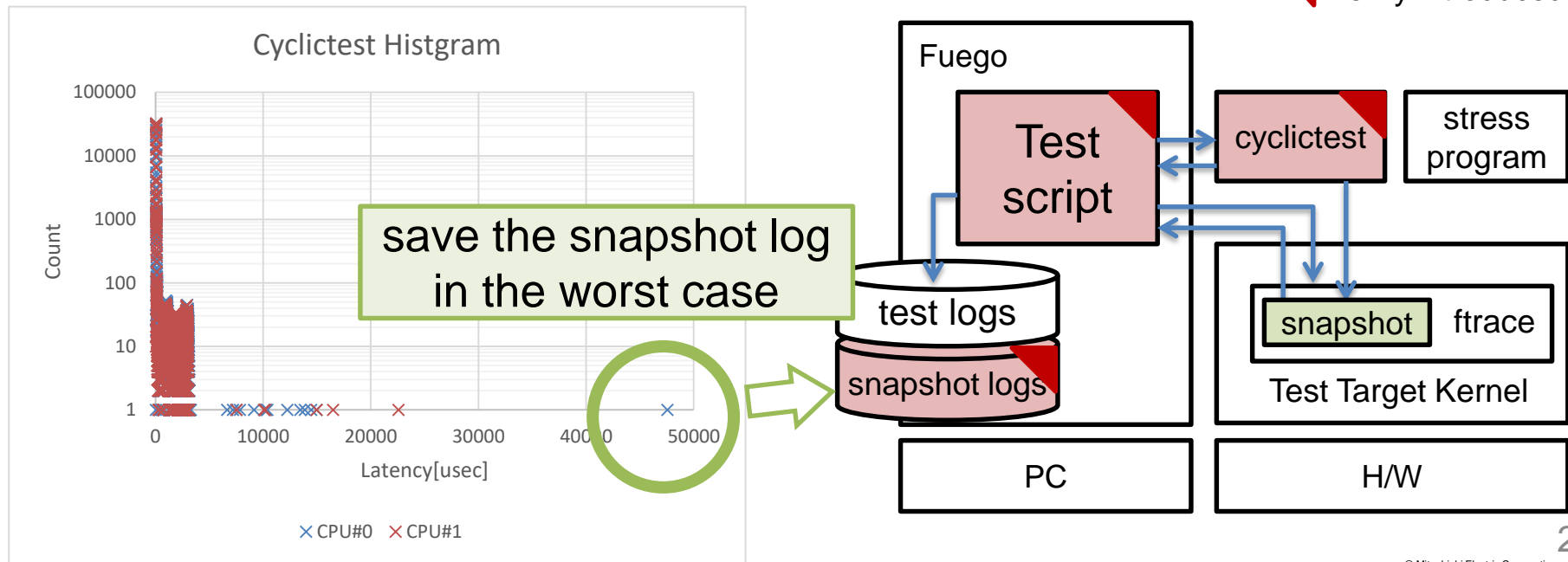
stop
tracing

stop
testing

The record may not be the worst case... k);

Improved Approach

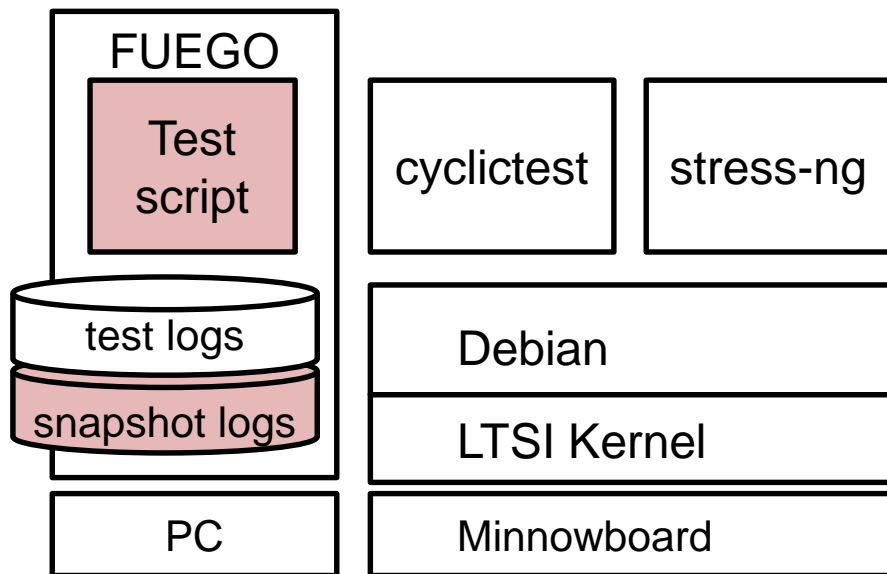
- Using a ftrace feature "Snapshot"
 - Cyclictest is modified to take a snapshot when maximum-latency is updated.



OUR ACTUAL CASE WITH AN IMPROVED APPROACH...

Detail of Our Evaluation Environment

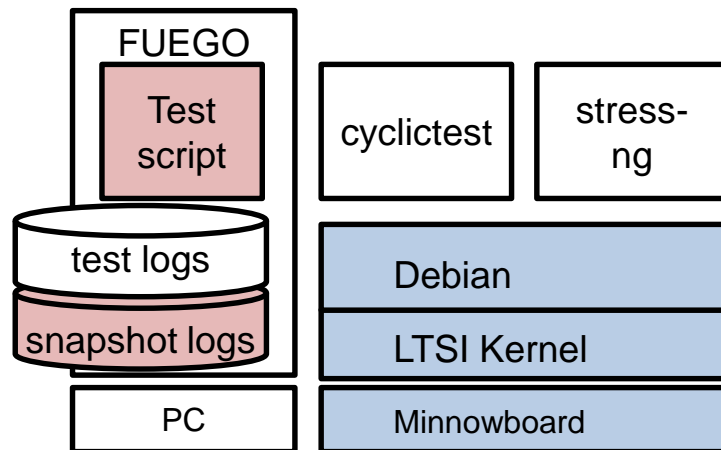
- Our test case
 - Latency of real-time priority process under some kinds of stress by non-realtime process.



- Cyclictest with realtime priority
- Stress-ng with non-realtime priority
 - making stressful

Hardware and OS

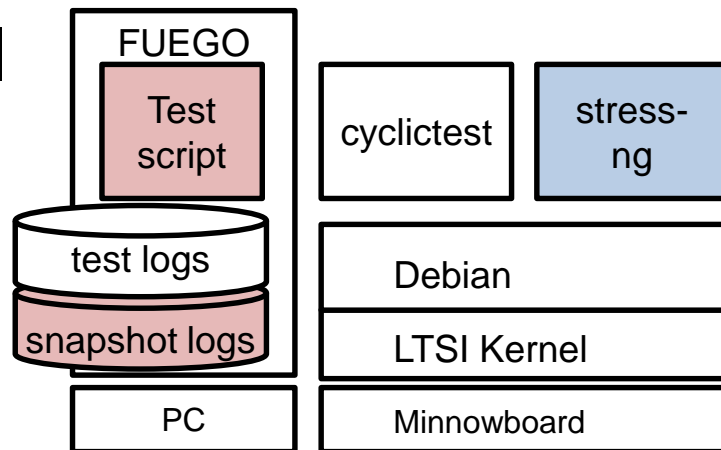
- Target board: Minnowboard Turbot Dual-core
 - Intel Atom E3826
 - #Cores / Threads: 2/2
 - Freq / Cache : 1.46GHz / 1MB
 - 2GB DDR3L 1067MT/s
 - Storage: microSD
 - Ethernet: Intel i211
- Debian Gnu/Linux 9.5.0
 - Linux 4.14 LTSI



Stress-ng

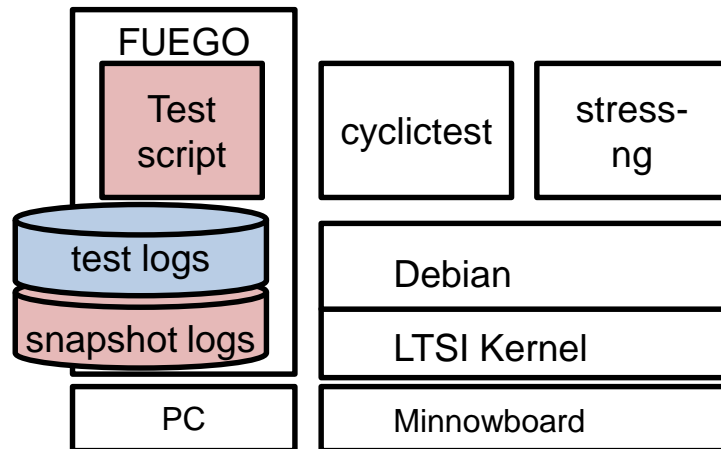
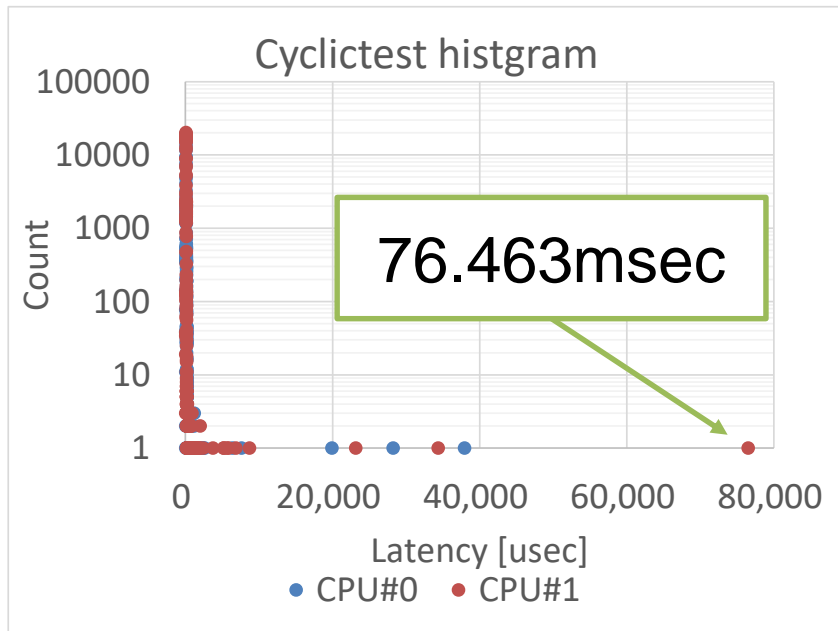
- stress-ng has stressors for a lot of components
 - cpu, fork, timer, sync, dentry, flock, udp, pipe, semaphore...
 - Beware, extreme scenarios seldom happen in real-life.

- `$./stress-ng --stressors | wc -l`
207



Result

- Stress-ng
 - Create&delete directory entry (dentry), 8 instances without CPUSET, with non-realtime priority.
 - 1 hour



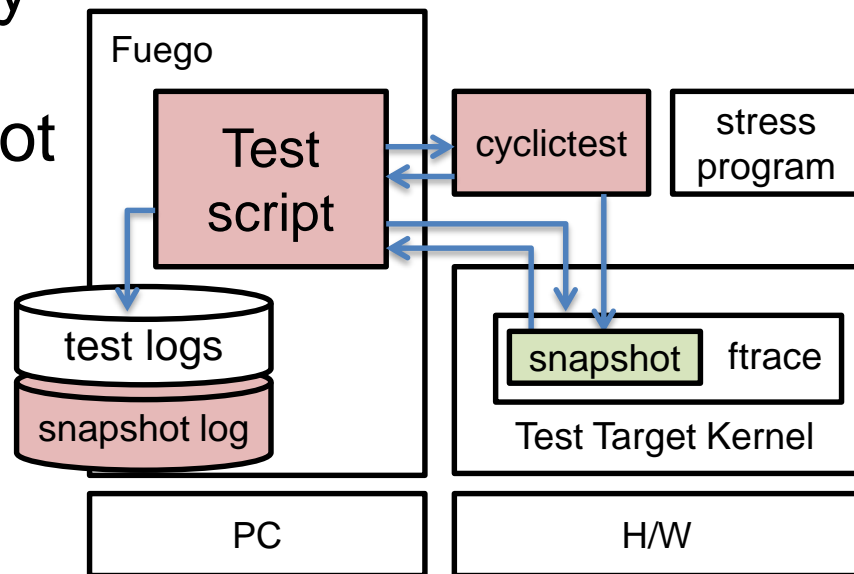
Result

- Snapshot log in the worst case.

```
cyclictest-17361 [001] .... 55024.413733: mutex_lock <-__fdget_pos
cyclictest-17361 [001] .... 55024.413733: vfs_write <-SyS_write
cyclictest-17361 [001] .... 55024.413734: rw_verify_area <-vfs_write
cyclictest-17361 [001] .... 55024.413735: security_file_permission <-
    rw_verify_area
cyclictest-17361 [001] .... 55024.413736: __sb_start_write <-vfs_write
cyclictest-17361 [001] .... 55024.413737: __add <-
    76.463msec
cyclictest-17361 [001] ...1 55024.413738: __sub <-
    __sb_start_write
cyclictest-17361 [001] .... 55024.413739: __vfs_write <-vfs_write
cyclictest-17361 [001] .... 55024.413741: tracing_mark_write: hit latency
    snapshot threshold (76463 > 1000)
cyclictest-17361 [001] .... 55024.413745: __fsnotify_parent <-vfs_write
cyclictest-17361 [001] .... 55024.413746: fsnotify <-vfs_write
```

Evaluation

- We got clues to detect the factor by doing a test and tracing at the same time.
- Fuego helped repeat execution of both.
- Therefore, we can effectively figure out the reason of the delay with using the snapshot log.



- Who I am
- Overview
- Related Tools
 - Automated Test Framework / Fuego
 - Real-time latency tool / cyclicttest
 - Tracing kernel feature / ftrace
- Issue
- Approach
 - Our Use Cases
- Conclusion and Future work

CONCLUSION

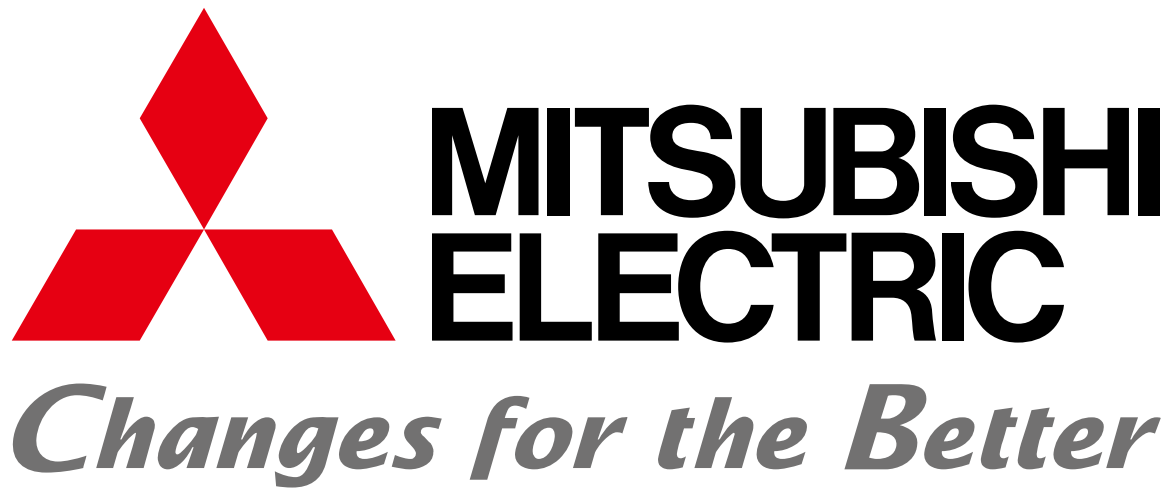
- Summary
 - It is important to ensure adequate performance before releasing products.
 - It is necessary to repeat tests for reproducing the rare case which does not meet real-time performance requirements.
 - Fuego is useful to us for not only measuring but also tracing at the same time.
- Future Works
 - Discussion with Fuego community for adding our test script which I have shown.

Resources

- Test Framework: FUEGO
 - <http://fuegotest.org/>
 - <https://elinux.org/Fuego>
- LTSI Project
 - <https://ltsi.linuxfoundation.org/>
- AGL Test framework: AGL-JTA
 - <https://wiki.automotivelinux.org/agl-jta>
- rt-tests
 - <https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/rt-tests>
- stress-ng
 - <http://kernel.ubuntu.com/~cking/stress-ng/>

THANK YOU!

Any Questions?



APPENDIX

Use Case: Cyclictest Options

- 1 msec latency, 10 msec interval, in 1 hour.
 - \$ cyclictest
 - --histogram=10000 --interval=10000 --duration=3600s
 - --smp --quiet --mlockall --priority=50
 - --snapshot=1000 (instead of --breakevent)

Use Case: Test Script

- Test script runs on target.
 - 1: chrt itself
 - 2: run stress-ng program as non-rt process
 - 3: maximize /proc/sys/kernel/sched_rt_runtime_us
 - 4: setup ftrace configurations
 - echo 0 > \$ftracedir/tracing_on
 - echo 0 > \$ftracedir/snapshot
 - echo function > \$ftracedir/current_tracer
 - echo 1 > \$ftracedir/tracing_on
 - 5: run cyclicttest
 - 6: save the log and a snapshot data (if recorded) to Fuego
 - 7: normalize /proc/sys/kernel/sched_rt_runtime_us